In the Claims:

Please amend claims 1, 4, 11, 19, 21, 24, 28, 33, 35, 36, 39-42, 47, 51, 53, 56, 57 and 60, and please add claims 63-66, as indicated below.

1. (Currently Amended) A method for reloading classes in an application, the method comprising:

a class loader, in response to an invocation from a class loader controller, loading a class in the application, wherein the class loader is one of a hierarchal stack of class loaders each configured to load one or more classes in the application, wherein the class loader controller provides an interface to the hierarchal stack of class loaders and a common entry point for loading classes of the application;

detecting the class has been changed;

the class loader controller replacing the class loader in the hierarchy of class loaders with a new class loader for the detected changed class; and

the new class loader reloading the changed class in the application;

wherein said detecting the class has been changed, said replacing the class loader and said reloading the changed class are performed while the application is executing.

2. (Original) The method as recited in claim 1, wherein only the changed class and other classes with dependencies on the changed class are reloaded in response to said detecting the class has been changed.

3. (Original) The method as recited in claim 1, wherein said detecting the class has been changed, said replacing the class loader and said reloading the changed class are performed without restarting the application.

4. (Currently Amended) The method as recited in claim 1, further comprising:

the [[a]] class loader controller receiving a request to load the class prior to the class loader loading the class; and

the class loader controller determining that the class loader in the hierarchal stack of class loaders is responsible for loading the class~~; and~~

~~the class loader controller invoking the class loader to perform said loading the class in the application~~.

5. (Original) The method as recited in claim 1, further comprising:

registering the loaded class with a dirty class monitor; and

the dirty class monitor performing said detecting the class has been changed.

6. (Original) The method as recited in claim 5, further comprising:

the dirty class monitor notifying a class loader controller that the class has been changed;

the class loader controller performing said replacing the class loader in the hierarchy of class loaders with the new class loader for the class in response to said notification; and

the class loader controller invoking the new class loader to perform said reloading the changed class in the application.

7. (Original) The method as recited in claim 1, further comprising:

determining one or more classes with dependencies on the changed class;

replacing one or more class loaders in the hierarchy of class loaders, wherein the one or more class loaders are each configured to load one or more of the one or more classes with dependencies on the changed class; and

the one or more class loaders each reloading the one or more classes in the application with dependencies on the changed class which the particular class loader is configured to load, wherein said reloading is performed while the application is executing.

8. (Original) The method as recited in claim 1, wherein the application is one of a plurality of applications executing within an application server, wherein each of the one or more applications is associated with an application-specific hierarchy of class loaders configured to load classes in the particular application.

9. (Original) The method as recited in claim 8, wherein the application-specific hierarchy of class loaders in each application is configured to load the classes in the particular application while the particular application is executing.

10. (Original) The method as recited in claim 8, wherein each of the class loaders in the application-specific hierarchy of class loaders in each application is configured to be replaced to reload one or more changed classes in the particular application while the particular application is executing.

11. (Currently Amended) The method as recited in claim 8, wherein the application server is based on ~~the~~ a Java™ 2 Platform, Enterprise Edition ~~(J2EE™)~~ specification.

12. (Original) The method as recited in claim 1, wherein the application comprises one or more modules, wherein the hierarchical stack of class loaders includes a module class loader for each module in the application, and wherein the module class loader associated with a particular module is configured to load one or more classes of the particular module.

13. (Original) The method as recited in claim 12, wherein the hierarchical stack of class loaders further includes an application class loader, wherein the application class loader is the parent class loader of the one or more module class loaders in the hierarchical stack of class loaders.

14. (Original) The method as recited in claim 13, wherein the application class loader is configured to load utility classes used in the application.

15. (Original) The method as recited in claim 13, wherein the application class loader is configured to load classes used by more than one module in the application.

16. (Original) The method as recited in claim 13, wherein the application is executing within an application server, wherein the application server includes a system class loader, wherein the system class loader is the parent class loader of the application class loader in the hierarchical stack of class loaders.

17. (Original) The method as recited in claim 16, wherein the system class loader is configured to load standard classes.

18. (Original) The method as recited in claim 16, wherein the system class loader is configured to load core classes of the application server.

19. (Currently Amended) The method as recited in claim 12, wherein the stack of class loaders further includes one or more class loaders based on an Enterprise JavaBeans (EJB) specification ~~class loaders~~, wherein each of the one or more ~~EJB™~~ class loaders is a child of one module class loader in the hierarchical stack of class loaders.

20. (Original) The method as recited in claim 12, wherein the stack of class loaders further includes one or more Web class loaders, wherein each of the one or more Web class loaders is a child of one module class loader in the hierarchical stack of class loaders.

21. (Currently Amended) A method for dynamically reloading classes in an application executing within an application server, the method comprising:

changing a class used by the application;

a class loader controller replacing a class loader for the class in the application with a new class loader for the changed class, wherein the class loader is one of a hierarchal stack of class loaders each configured to load one or more classes in the application, wherein the class loader controller provides an interface to the hierarchal stack of class loaders and a common entry point for loading classes of the application;

the class loader controller replacing one or more class loaders for one or more classes with dependencies on the changed class, wherein the one or more class loaders are included in the hierarchical stack of class loaders;

the new class loader, in response to an invocation from the class loader controller, reloading the changed class; and

the replaced one or more class loaders reloading the one or more classes in the application with dependencies on the changed class;

wherein only the class loaders for the changed class and the one or more classes with dependencies on the changed class are replaced in response to said changing the class; and

wherein said replacing the class loaders and said reloading the classes are performed while the application is executing without restarting the application.

22. (Original) The method as recited in claim 21, wherein the application comprises one or more modules, wherein each module in the application is associated with a module class loader for the particular module configured to load one or more classes of the particular module, and wherein the one or more module class loaders are included in the hierarchical stack of class loaders.

23. (Original) The method as recited in claim 22, wherein the hierarchical stack of class loaders includes an application class loader configured to load classes used by more than one module in the application, and wherein the application class loader is the parent class loader of the one or more module class loaders in the hierarchical stack of class loaders.

24. (Currently Amended) The method as recited in claim 23, wherein the hierarchical stack of class loaders further includes one or more class loaders based on an Enterprise JavaBeans (EJB) specification ~~class loaders~~, wherein each ~~EJB™~~ class loader is a child of one module class loader in the hierarchical stack of class loaders.

25. (Original) The method as recited in claim 23, wherein the hierarchical stack of class loaders further includes one or more Web class loaders, wherein each Web class loader is a child of one module class loader in the hierarchical stack of class loaders.

26. (Original) The method as recited in claim 23, wherein the application server includes a system class loader configured to load core classes of the application server, wherein

the system class loader is the parent class loader of the application class loader in the hierarchical stack of class loaders.

27. (Original) The method as recited in claim 21, wherein the application server is operable to execute a plurality of applications, wherein each application includes a hierarchical stack of class loaders configured to load classes for the particular application.

28. (Currently Amended) The method as recited in claim 21, wherein the application server is based on ~~the~~ a Java™ 2 Platform, Enterprise Edition ~~(J2EE™)~~ specification.

29. (Original) A system comprising:

a processor;

a memory operable to store program instructions, wherein the program instructions implement an application server executable by the processor within the system, wherein the program instructions further implement a plurality of applications executable by the processor within the system;

wherein the application server is operable to provide access to the plurality of applications to clients of the application server;

wherein one or more of the plurality of applications each includes a dynamic class reloading module comprising a hierarchical stack of class loaders, wherein the hierarchical stack of class loaders includes a separate class loader for each module in the particular application, and wherein each class loader is operable to reload one or more classes used by the particular application;

wherein, for each of the one or more applications, the dynamic class reloading modules is operable during execution of the application to:

detect that a class used by the application has been changed;

replace a class loader for the class in the hierarchical stack of class loaders with a new class loader for the detected changed class; and

wherein the new class loader is operable to reload the changed class in the first application during execution of the first application.

30. (Original) The system as recited in claim 29, wherein said detecting, said replacing and said reloading are performed without restarting the application.

31. (Original) The system as recited in claim 30, wherein said detecting, said replacing and said reloading are performed without restarting the application server.

32. (Original) The system as recited in claim 29, wherein, for each of the one or more applications, the dynamic class reloading modules is further operable during execution of the application to replace one or more other class loaders in response to said detecting, wherein the one or more other class loaders are operable to reload one or more classes with dependencies on the changed class.

33. (Currently Amended) The system as recited in claim 29, wherein ~~each dynamic class reloading module further comprises a class loader controller~~ wherein the class loader controller of each dynamic class reloading module is operable to:

receive a notification that the class has been changed;

determine which class loader in the hierarchical stack of class loaders is operable to load the class;

perform said replacing the class loader with the new class loader; and

invoke the new class loader to perform said reloading the changed class.

34. (Original) The system as recited in claim 33, wherein each dynamic class reloading module further comprises a dirty class monitor operable to:

perform said detecting that the class used by the application has been changed; and

notify the class loader controller that the class has been changed.

35. (Currently Amended) The system as recited in claim 29, wherein the application server is based on ~~the~~ a Java™ 2 Platform, Enterprise Edition ~~(J2EE™)~~ specification.

36. (Currently Amended) A system comprising:

a processor;

a memory operable to store program instructions, wherein the program instructions implement an application executable by the processor within the system, wherein the application includes a dynamic class reloading module comprising:

a hierarchical stack of class loaders, wherein each of the hierarchical stack of class loaders is executable to load one or more classes in the application;

a class loader controller operable to:

provide an interface to the hierarchal stack of class loaders;

provide a common entry point for loading classes of the application;

~~wherein the application is executable within the system to:~~

invoke a class loader to load a class in the application, wherein the class loader is one of the hierarchal stack of class loaders;

detect the class has been changed;

replace the class loader in the hierarchy of class loaders with a new class loader for the detected changed class; and

invoke the new class loader to reload the changed class in the application;

wherein said detecting the class has been changed, said replacing the class loader and said reloading the changed class are performed while the application is executing.

37. (Original) The system as recited in claim 36, wherein only the changed class and other classes with dependencies on the changed class are reloaded in response to said detecting the class has been changed.

38. (Original) The system as recited in claim 36, wherein said detecting the class has been changed, said replacing the class loader and said reloading the changed class are performed without restarting the application.

39. (Currently Amended) The system as recited in claim 36, wherein the ~~application further includes a~~ class loader controller is executable within the application to:

receive a request to load the class prior to the class loader loading the class;

determine that the class loader is responsible for loading the class; and

perform said invoking the class loader to load the class in the application.

40. (Currently Amended) The system as recited in claim 36, wherein the ~~application further includes a~~ class loader controller is executable within the application to perform said detecting the class has been changed.

41. (Currently Amended) The system as recited in claim 40, wherein the ~~application further includes a~~ class loader controller is executable within the application to:

receive notification from the dirty class monitor that the class has been changed;

perform said replacing the class loader in the hierarchy of class loaders with the new class loader for the class in response to said receiving notification; and

perform said invoking the new class loader to reload the changed class in the application.

42. (Currently Amended) The system as recited in claim 36, wherein the ~~application further includes a~~ class loader controller is executable within the application to:

receive notification that the class has been changed;

perform said replacing the class loader in the hierarchy of class loaders with the new class loader for the class in response to said receiving notification; and

perform said invoking the new class loader to reload the changed class in the application.

43. (Original) The system as recited in claim 36, wherein the application is further executable within the system to:

determine one or more classes with dependencies on the changed class;

replace one or more class loaders in the hierarchy of class loaders, wherein the one or more class loaders are each configured to load one or more of the one or more classes with dependencies on the changed class; and

invoke each of the one or more class loaders to reload the one or more classes in the application with dependencies on the changed class which the particular class loader is configured to load.

44. (Original) The system as recited in claim 36, wherein the program instructions further implement an application server executable within the system and a plurality of applications executable within the application server, wherein the application is one of the plurality of applications, wherein each of the one or more applications is associated with an application-specific hierarchy of class loaders configured to load classes in the particular application.

45. (Original) The system as recited in claim 44, wherein each of the plurality of applications is executable within the application server to invoke one or more of the hierarchy of class loaders to load the classes in the particular application while the particular application is executing within the application server.

46. (Original) The system as recited in claim 45, wherein each of the plurality of applications is executable within the application server to:

replace one or more of the class loaders in the application-specific hierarchy of class loaders in the particular application; and

invoke each of the replaced one or more class loaders to reload one or more changed classes in the particular application while the particular application is executing.

47. (Currently Amended) The system as recited in claim 45, wherein the application server is based on ~~the~~ a Java™ 2 Platform, Enterprise Edition ~~(J2EE™)~~ specification.

48. (Original) The system as recited in claim 36, wherein the application comprises one or more modules, wherein the hierarchical stack of class loaders includes a module class loader for each module in the application, and wherein the module class loader associated with a particular module is configured to be invoked by the application to load one or more classes of the particular module.

49. (Original) The system as recited in claim 48, wherein the hierarchical stack of class loaders further includes an application class loader, wherein the application class loader is the parent class loader of the one or more module class loaders in the hierarchical stack of class loaders, and wherein the application class loader is configured to be invoked by the application to load classes used by more than one module in the application.

50. (Original) The system as recited in claim 49, wherein the application is executing within an application server, wherein the application server includes a system class loader, wherein the system class loader is the parent class loader of the application class loader in the hierarchical stack of class loaders, and wherein the system class loader is configured to be invoked by the application to load core classes of the application server.

51. (Currently Amended) The system as recited in claim 48, wherein the stack of class loaders further includes one or more class loaders based on an Enterprise JavaBeans

(EJB) specification ~~class loaders~~, wherein each of the one or more ~~EJB™~~ class loaders is a child of one module class loader in the hierarchical stack of class loaders.

52. (Original) The system as recited in claim 48, wherein the stack of class loaders further includes one or more Web class loaders, wherein each of the one or more Web class loaders is a child of one module class loader in the hierarchical stack of class loaders.

53. (Currently Amended) A carrier medium comprising program instructions, wherein the program instructions are computer-executable to implement:

a class loader, in response to an invocation from a class loader controller, loading a class in the application wherein the class loader is one of a hierarchal stack of class loaders each configured to load one or more classes in the application, wherein the class loader controller provides an interface to the hierarchal stack of class loaders and a common entry point for loading classes of the application;

detecting the class has been changed;

the class loader controller replacing the class loader in the hierarchy of class loaders with a new class loader for the detected changed class; and

the new class loader reloading the changed class in the application;

wherein said detecting the class has been changed, said replacing the class loader and said reloading the changed class are performed while the application is executing.

54. (Original) The carrier as recited in claim 53, wherein only the changed class and other classes with dependencies on the changed class are reloaded in response to said detecting the class has been changed.

55. (Original) The carrier medium as recited in claim 53, wherein said detecting the class has been changed, said replacing the class loader and said reloading the changed class are performed without restarting the application.

56. (Currently Amended) The carrier medium as recited in claim 53, wherein the program instructions are further computer-executable to implement:

[[a]] the class loader controller receiving a request to load the class prior to the class loader loading the class; and

the class loader controller determining that the class loader is responsible for loading the class~~; and~~

~~the class loader controller invoking the class loader to perform said loading the class in the application~~.

57. (Currently Amended) The carrier medium as recited in claim 53, wherein the program instructions are further computer-executable to implement:

notifying [[a]] the class loader controller that the class has been changed;

the class loader controller performing said replacing the class loader in the hierarchy of class loaders with the new class loader for the class in response to said notification; and

the class loader controller invoking the new class loader to perform said reloading the changed class in the application.

58. (Original) The carrier medium as recited in claim 53, wherein the program instructions are further computer-executable to implement:

determining one or more classes with dependencies on the changed class;

replacing one or more class loaders in the hierarchy of class loaders, wherein the one or more class loaders are each configured to load one or more of the one or more classes with dependencies on the changed class; and

the one or more class loaders each reloading the one or more classes in the application with dependencies on the changed class which the particular class loader is configured to load, wherein said reloading is performed while the application is executing.

59. (Original) The carrier medium as recited in claim 53, wherein the application is one of a plurality of applications executing within an application server, wherein each of the one or more applications is associated with an application-specific hierarchy of class loaders configured to load classes in the particular application, wherein each of the class loaders in the application-specific hierarchy of class loaders in each application is configured to be replaced to reload one or more changed classes in the particular application while the particular application is executing.

60. (Currently Amended) The carrier medium as recited in claim 59, wherein the application server is based on ~~the~~ [[a]] Java™ 2 Platform, Enterprise Edition ~~(J2EE™)~~ specification.

61. (Original) The carrier medium as recited in claim 53,

wherein the application comprises one or more modules, wherein the hierarchical stack of class loaders includes a module class loader for each module in

the application, and wherein the module class loader associated with a particular module is configured to load one or more classes of the particular module;

wherein the hierarchical stack of class loaders further includes an application class loader, wherein the application class loader is the parent class loader of the one or more module class loaders in the hierarchical stack of class loaders, and wherein the application class loader is configured to load classes used by more than one module in the application.

62. (Original) The carrier medium as recited in claim 61, wherein the application is executing within an application server, wherein the application server includes a system class loader, wherein the system class loader is the parent class loader of the application class loader in the hierarchical stack of class loaders, wherein the system class loader is configured to load core classes of the application server.

63. (New) A class loader module executable within an application executing within an application server, comprising:

a hierarchal stack of class loaders each configured to load one or more classes in the application;

a class loader controller configured to provide an interface to a hierarchal stack of class loaders and a common entry point for loading classes of the application;

a dirty class monitor operable to:

detect that a class used by the application has been changed; and

notify the class loader controller that the class has been changed.

64. (New) The class loader module as recited in claim 63, wherein the class loader controller is configured to:

receive a notification from the dirty class monitor that the class has been changed;

determine a class loader in the hierarchal stack of class loaders operable to load the changed class;

replace the class loader in the hierarchal stack of class loaders with a new class loader for the changed class, wherein the new class loader is operable to reload the changed class in the application during execution of the application; and

invoke the new class loader to reload the changed class.

65. (New) The class loader module as recited in claim 64, wherein said detecting, said notifying and said replacing are preformed without restarting the application or the application server.

66. (New) The class loader module as recited in claim 64, wherein the class loader controller is further configured to replace one or more other class loaders is response to said receiving, wherein the one or more other class loaders are operable to reload one or more classes with dependencies on the changed class.